Document Coding Style

Echipa Farik

1. Sursele Java

Toate sursele Java ale proiectului vor fi fisiere text cu extensia **.java**.

2. Clasele Java

Un fisier .java va contine definitia unei singure clase. Se recomanda ca acesta clasa sa fie publica.

3. Comentarii

1. Comentariile vor fi in stilul C. De exemplu, fiecare clasa poate incepe cu un comentariu de genul urmator, dar nu obligatoriu:
```
/*
 * Classname
 *
 * Version info
 *
 * Copyright notice
 */
```
2. Comentariile pe o singura linie vor fi facute folosind **/* … */** si nu //. Exemplu:
```
if (condition) {
    /* Handle the condition. */
    ...
}
```
3. Daca exista *Trailing Comments*, acestea vor fi suficient de mult departate de codul pe care il comenteaza, de exemplu:
```
if (a == 2) {
    return TRUE;           /* special case */
} else {
    return isP(a);         /* works only for odd a */
}
```

4. Comentariile de documentatie se fac folosind **/** … */**. Fiecare comentariu de documentatie poate contine tag-uri specifice comentariilor de documentatie: @return, @param, @version, @author.

4. Pachete si import-uri

In general, prima linie necomentata va fi o declaratie de pachet, daca exista. Apoi se vor declara import-urile. De exemplu,

```
package java.awt;
import java.awt.peer.CanvasPeer;
```

Daca exista mai multe clase importate din acelasi pachet, atunci importul se va realiza folosind *. De exemplu,

```
import java.awt.*;
```

5. Declararea claselor si a interfetelor

1. Fiecare clasa sau interfata va incepe cu un comentariu de documentatie **/** … */**.
2. Urmeaza declaratia de clasa sau interfata folosind **class** sau **interface**.
3. Urmeaza un comentariu de implementare **/* … */** pentru clasa, daca este cazul.
4. Se declara membri statici ai clasei. Mai intai cei **publi**c, apoi cei **protected** si la urma cei **private**.
5. Se declara membri de instanta ai clasei. Mai intai cei **public**e, apoi cei **protected** si la urma cei **private**.
6. Pentru punctele anterioare (4) si (5), ordinea va tine cont si de modificatorul final. Vor fi declarate mai intai variabilele cu modificatorul **final** si mai apoi variabilele fara modificatorul **final**.
7. Se declara constructorii.
8. Se declara metodele.

6. Indentarea surselor

Sursele vor fi indentate cu **tab**-uri formate din **4** spatii.

7. Lungimea unei linii

Liniile de cod nu vor avea mai mult de **80** de caractere.

8. Declararea membrilor

Fiecare membru al unei clase se va declara pe o linie distincta. Exemplu,

```
int level; // indentation level
int size;  // size of table
```

9. Initializarea membrilor

Variabilele locale se vor initializa acolo unde au fost declarate. Deci, si obiectele declarate ca variabile locale ale unei metode se vor instantia acolo unde au fost declarate.

10. Instructiuni

1. **Intructiunea simpla**
   Fiecare linie de cod va contine o singura declaratie/instructiune Java. Se vor evita intructiuni de genul: `x++; y--;`

2. **Instructiunea return**
   Valoarea intoarsa de return ar trebui sa fie scrisa intre paranteze doar daca face lucrurile mai clare. Exemple:

   ```
   return;
   return myDisk.size();
   return (size ? size : defaultSize);
   ```

3. **Instructiunile if, if-else, if-else-if-else**
   Instructiunea if va avea coding style-ul de mai jos; blocurile vor fi mereu delimitate de {}.

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else if (condition) {
    statements;
}
```

4. Instructiunea for

```
for (initialization; condition; update) {
    statements;
}
```

5. Instructiunea while

```
while (condition) {
    statements;
}

while (condition);
```

6. Instructiunea do-while

```
do {
    statements;
} while (condition);
```

7. try-catch-finally

```
try {
    statements;
} catch (ExceptionClass1 e) {
    statements;
}
catch (ExceptionClassN e) {
    statements;
}
finally {
    statements;
}
```

11. Conventii de numire

Clasele vor avea nume simple si sugestive. Numele fiecarei clasa va incepe cu litera mare. Daca exista clase cu nume format din mai multe cuvinte, atunci prima litera a cuvintelor care compun numele clasei va fi majuscula.

Numirea interfetelor se face la fel ca numirea claselor.

Numele metodelor vor incepe cu litera mica iar in cazul in care o metoda este formata din mai multe cuvinte, acestea vor fi scrise capitalizat (prima litera majuscula). Exemplu: `getBackgraound();`

Variabilele vor avea nume sugestive si vor incepe cu litera mica. Ca si in cazul metodelor, daca numele acestora este format din mai multe cuvinte, acestea – cuvintele – vor fi scrise capitalizat. Exemplu: `int myWidth;`

Constantele vor avea nume sugestive si vor fi scrise cu majuscula. Daca sunt formate din mai multe cuvinte, delimitatorul pentru cuvinte va fi underscore (_).

Numele pachetelor va fi format din cuvinte scrise cu litera mica. In cazul in care exista mai multe pachete acestea vor fi delimitate cu punct (.).

12. Liniile goale

Pentru a citi mai usor codul scris si pentru a delimita zone de cod relationate din punct de vedere logic, se vor folosi linii goale.

Lizibilitatea poate fi acoperita cu o linie goala:

- Intre metode
- Intre blocul variabilelor locale ale unei metode si prima intructiune din metoda
- Intre sectiuni logice ale unui bloc de cod

Nota: prin linie goala se intelege o linie blanc care nu contine tab-uri sau spatii.